Linear Prediction using NLMS algorithm

Theorem

Linear prediction is the estimation a signal by using previous value to estimate value that want to know. This method called Forward Linear Prediction

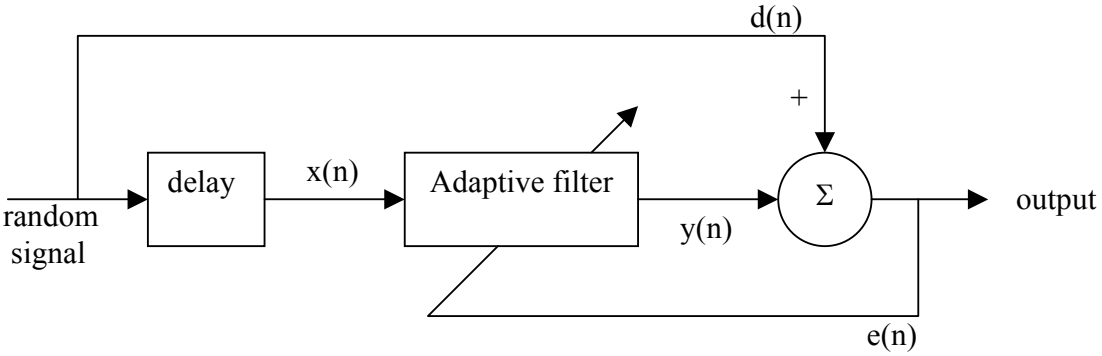


Figure 1 Block diagram of FIR linear prediction

The update equation of FIR adaptive filter using LMS algorithm

$$w(n+1) = w(n) + \mu(x(n)e^{*}(n)) \qquad \text{…….(1)}$$

The update equation of FIR adaptive filter using NLMS algorithm

$$\hat{w}(n+1) = \hat{w}(n) + \mu_v(x(n) / 1+\|x(n)\|_2^2)e^{*}(n) \qquad \text{…….(2)}$$

Source code

```
function varargout = linear(varargin)
% LINEAR Application M-file for linear.fig
%    FIG = LINEAR launch linear GUI.
%    LINEAR('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 21-Feb-2003 23:19:30

if nargin == 0  % LAUNCH GUI

	fig = openfig(mfilename,'reuse');

	% Generate a structure of handles to pass to callbacks, and store it.
	handles = guihandles(fig);
	guidata(fig, handles);

	if nargout > 0
		varargout{1} = fig;
	end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

	try
		if (nargout)
			[varargout{1:nargout}] = feval(varargin{:}); % FEVAL
switchyard
		else
			feval(varargin{:}); % FEVAL switchyard
		end
	catch
		disp(lasterr);
	end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
```

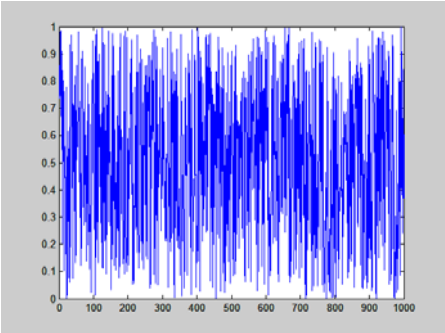
```
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.

% --------------------------------------------------------------------
function varargout = popupmenu1_Callback(h, eventdata, handles, varargin)
load x;
val = get(h,'Value');
switch val
case 1
        handles.data = 0;
case 2
        handles.data = rand(1000,1);
case 3
        handles.data = x;
end
guidata(h,handles)


% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)

% Get user input from GUI
x = handles.data;
```

```
L = str2double(get(handles.edit1,'String'));
d = str2double(get(handles.edit2,'String'));
u = str2double(get(handles.edit3,'String'));

% Calculate data
w = zeros(L,1);
x1 = zeros(L,1);
e(1:d-1)= x(1:d-1);
for n = d+1:size(x),
   x1 =[x(n-d);x1(1:L-1)] ;
   y(n) = (w)'*x1;
   e(n) = x(n) - y(n);
   c = x1.^2;
   c1= sum(c);
   w = w + (u/(1+c1))*x1*e(n);
end

% Create input plot
axes(handles.axes1)
plot(x)
set(handles.axes1,'XMinorTick','on')
grid on

% Create estimate plot
axes(handles.axes2)
plot(y)
set(handles.axes2,'XMinorTick','on')
grid on

% Create tap weight plot
axes(handles.axes3)
stem(w)
set(handles.axes3,'XMinorTick','on')
grid on

% Create error plot
axes(handles.axes4)
plot(e)
set(handles.axes4,'XMinorTick','on')
grid on
```

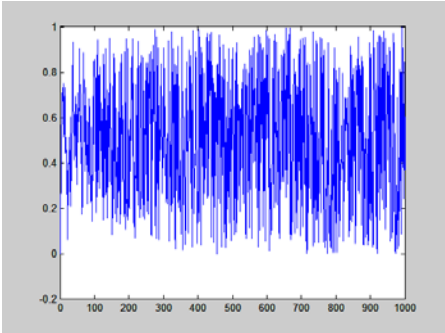
Simulation

Assume that input is random noise

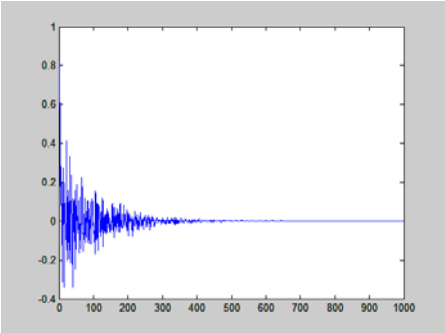
- delay = 0, μ = 0.5, L = 10



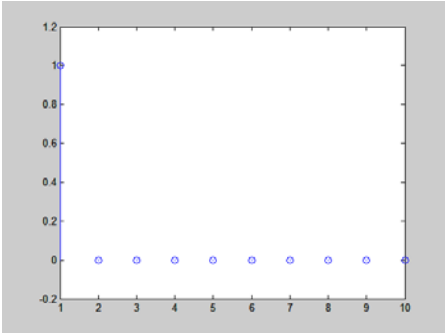
x(n)



y(n)

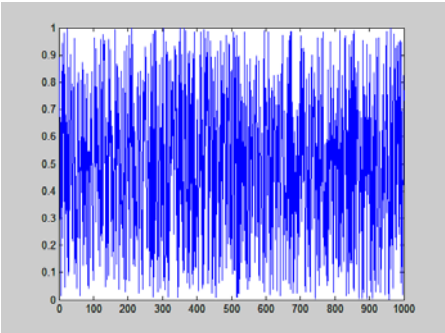


e(n)

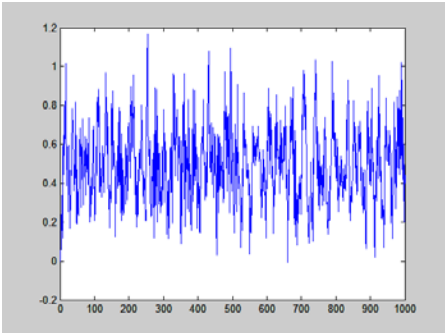


tap weight

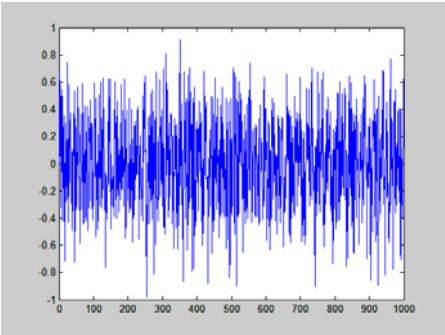
- delay = 1, μ = 0.5, L = 10



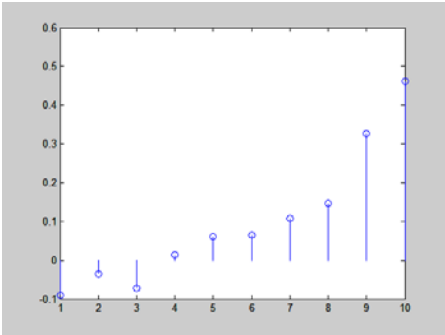
x(n)



y(n)

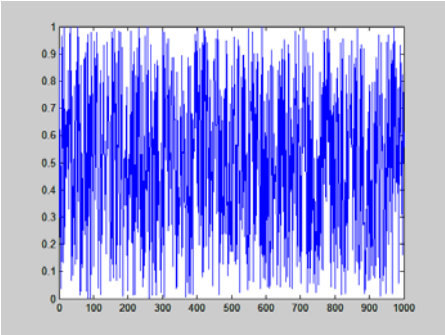


e(n)

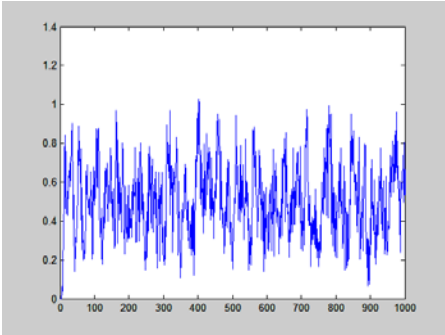


tap weight

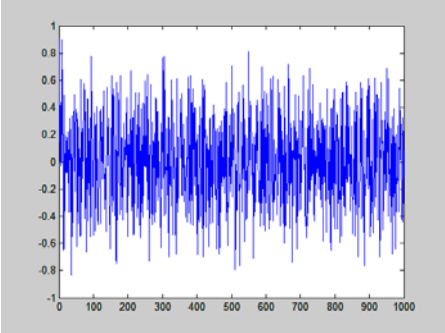
- delay = 3, μ = 0.5, L = 10



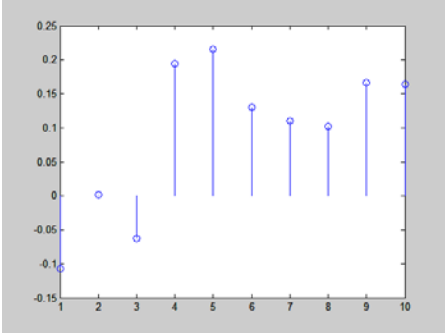
x(n)



y(n)



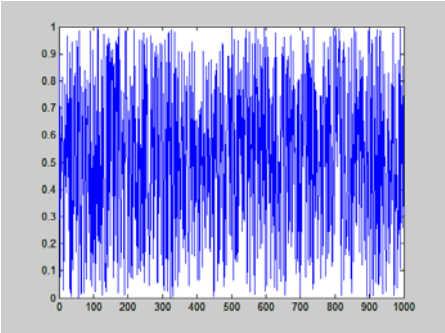
e(n)



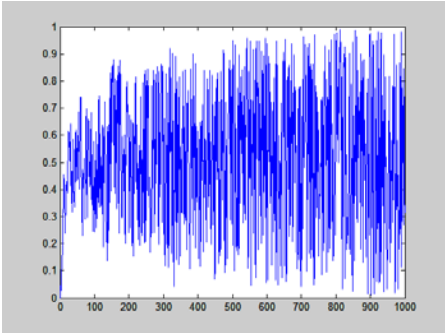
tap weight

From the result, if delay is changed, error will increase and y(n) will decrease. So that error is not converge.

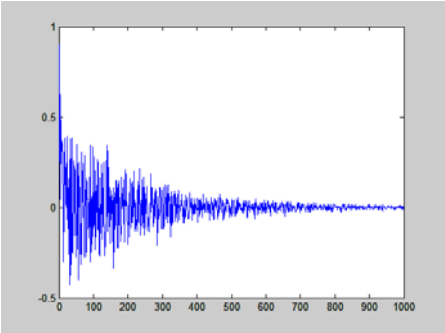
- μ = 0.2, delay = 0 , L = 10



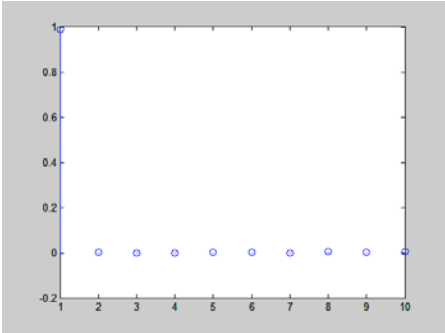
x(n)



y(n)

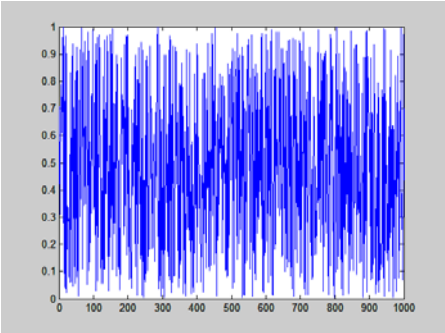


e(n)

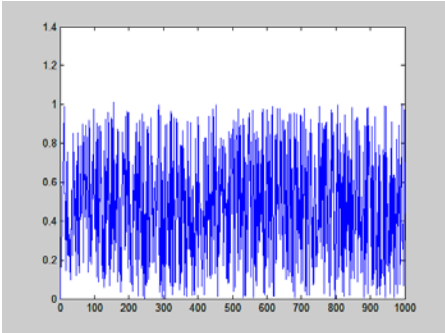


tap weight

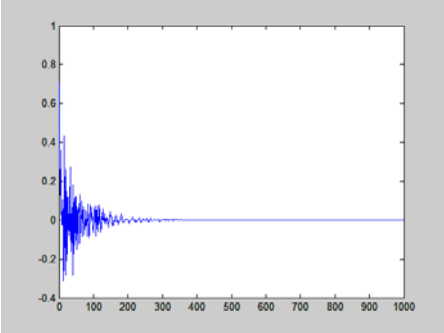
- μ = 0.8, delay = 0 , L = 10



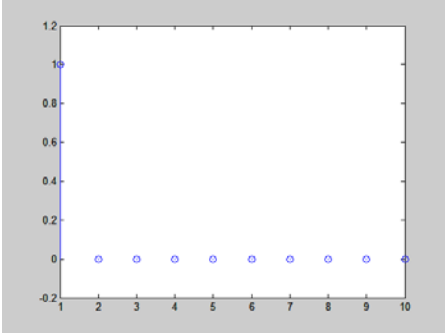
x(n)



y(n)



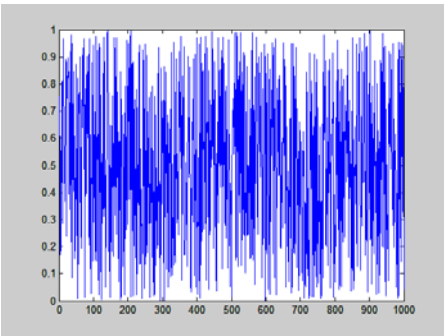
e(n)



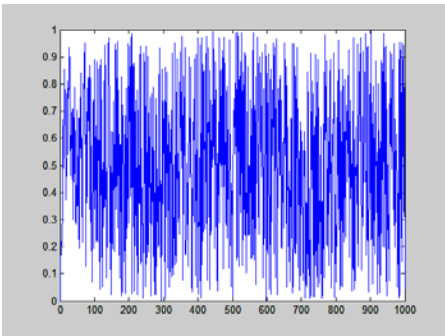
tap weight

From the result, if μ is changed, the convergence of signal will be changed too. If μ is a small number, it's a slow convergence. But if μ is a large number, it's a fast convergence

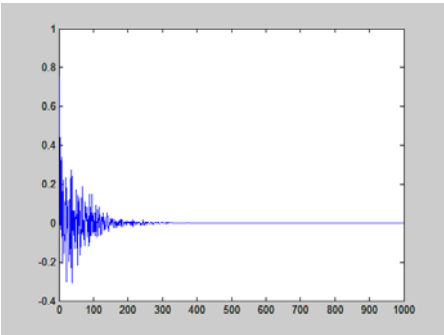
- L = 5, μ = 0.5, delay = 0



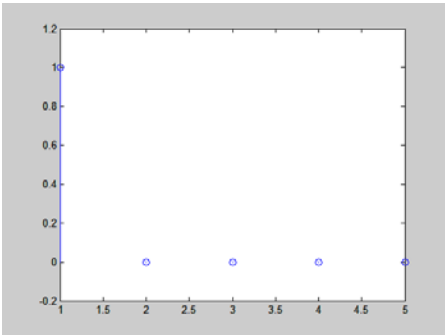
x(n)



y(n)

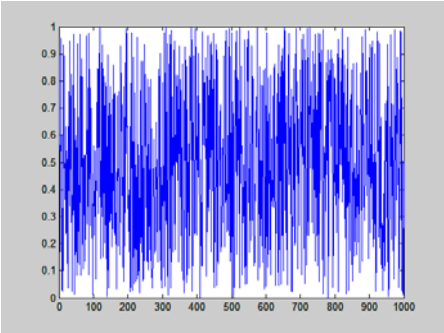


e(n)

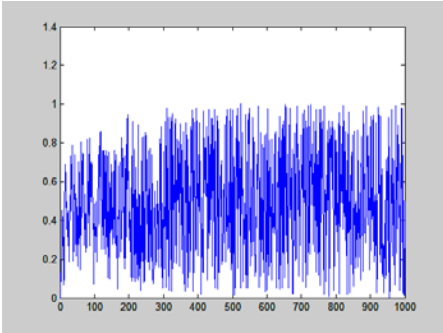


tap weight

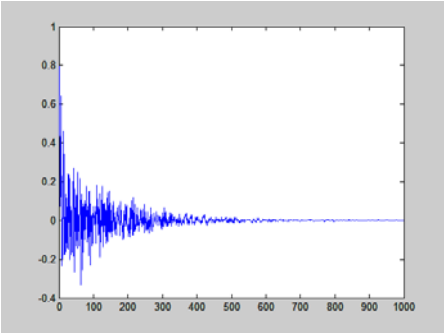
- L = 15, μ = 0.5, delay = 0



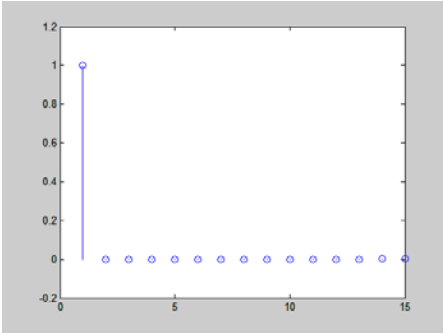
x(n)



y(x)



e(n)

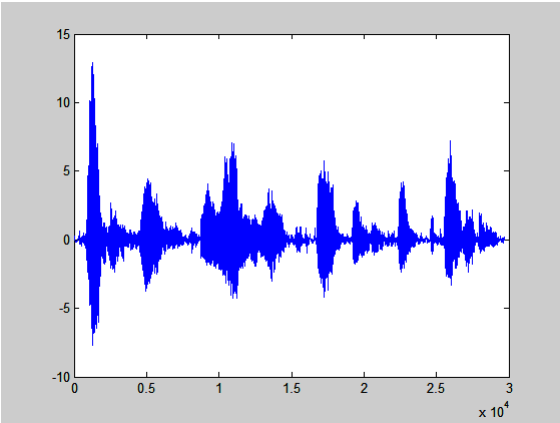


tap weight

From the result, if L is changed, the convergence of signal will be changed too. If L is a small number, it's a fast convergence. But if L is a large number, it's a slow convergence and have more complexity.

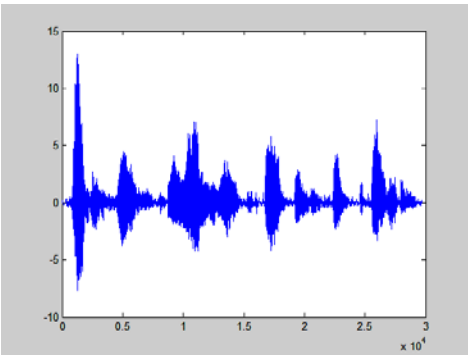
In case input is a speech signal.

Speech input waveform is as picture.

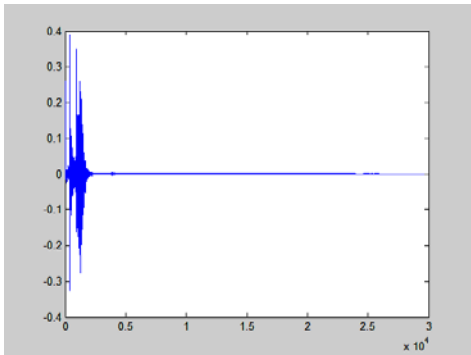


x(n)

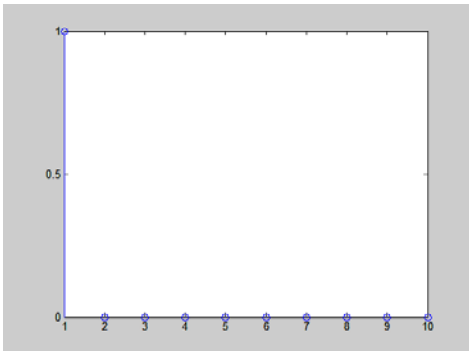
- delay = 0, μ = 0.5, L = 10



y(n)

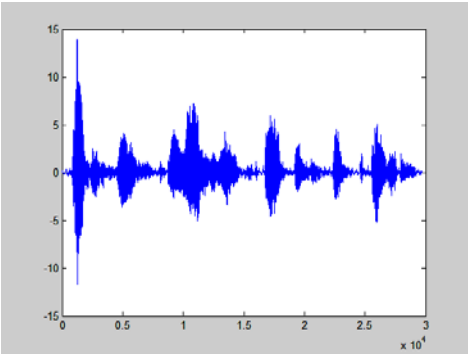


e(n)

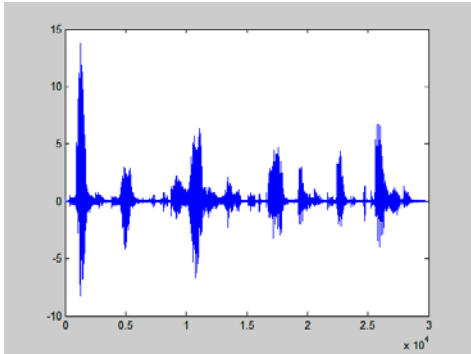


tap weight

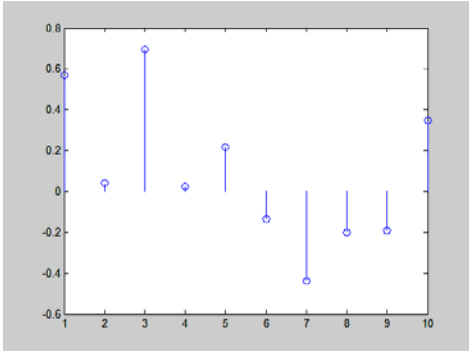
- delay = 1, μ = 0.5, L = 10



y(n)

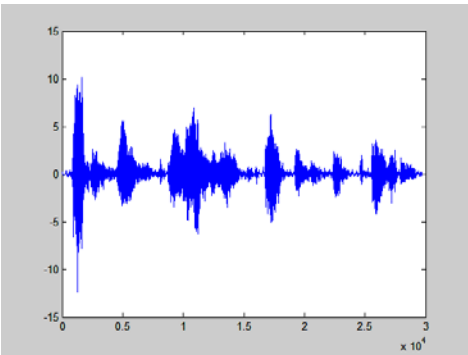


e(n)

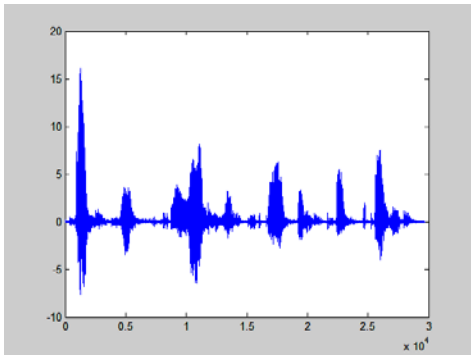


tap weight

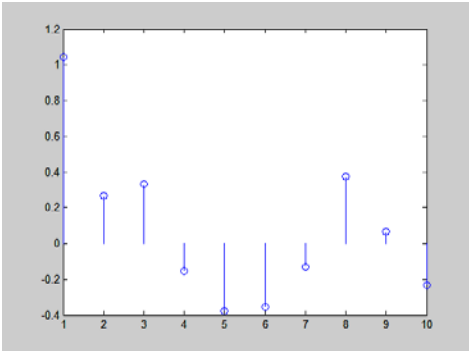
- delay = 3, μ = 0.5, L = 10



y(n)



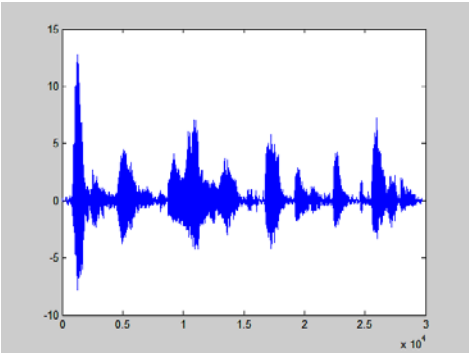
e(n)



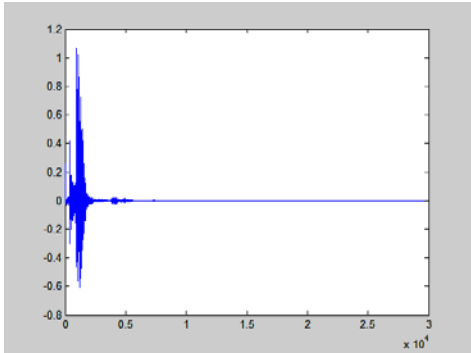
tap weight

From the result, if delay is zero, error will converge. But if delay is not zero, error will not converge. So that the linear prediction using NLMS algorithm cannot use for the system that have delay.

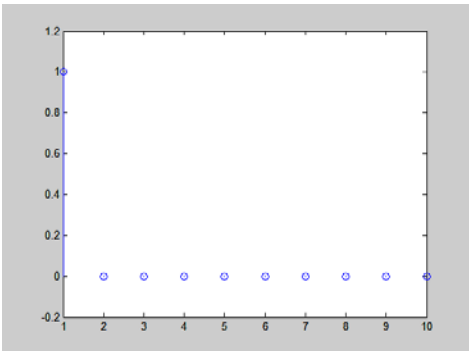
- μ = 0.2, delay = 0 , L = 10



y(n)

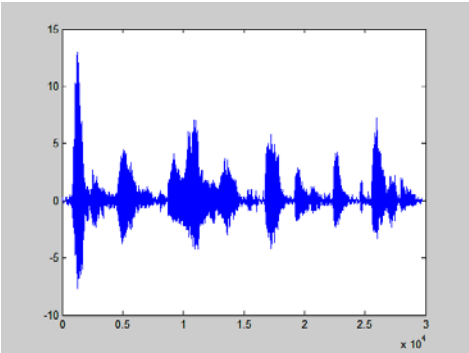


e(n)

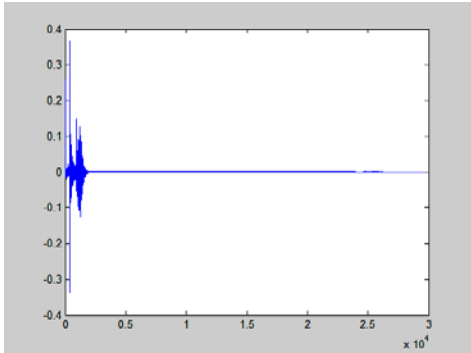


tap weight

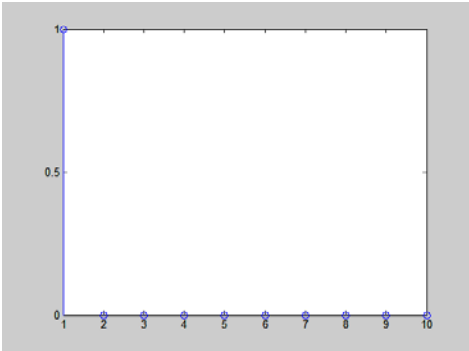
- μ = 0.8, delay = 0 , L = 10



y(n)



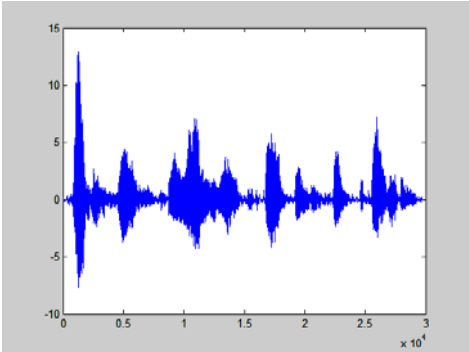
e(n)



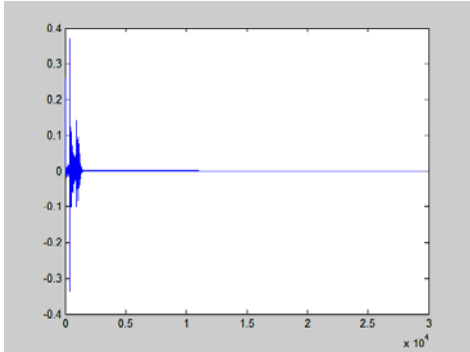
tap weight

From the result, if μ is changed, error will change. If μ is a large number, the system will have a small error. But if μ is a small number, the system will have more error.

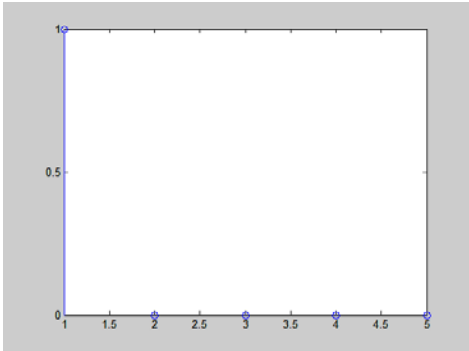
- L = 5, μ = 0.5, delay = 0



y(n)

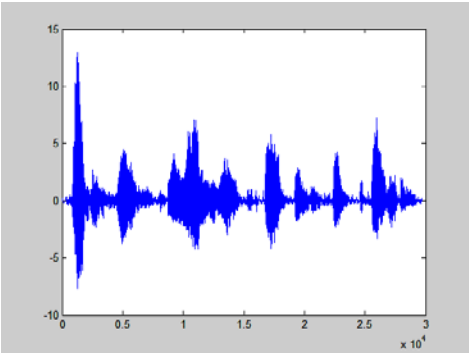


e(n)

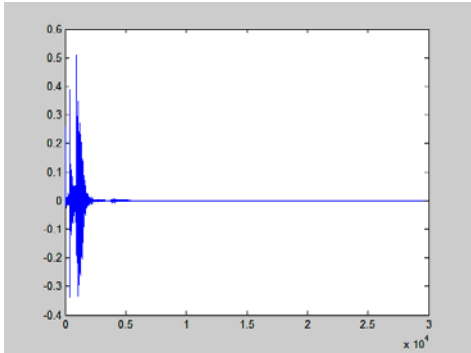


tap weight

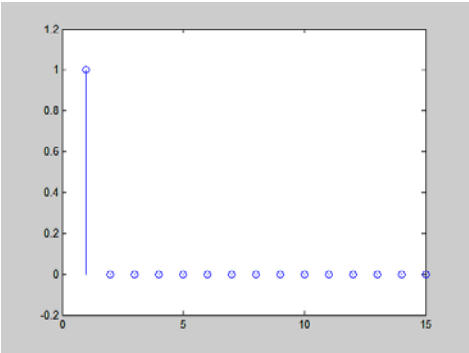
- L = 15, μ = 0.5, delay = 0



y(n)



e(n)



tap weight

From the result, if L is changed, error will change. If L is a small number, the system will have a small error. But if L is a large number, the system will have more error.

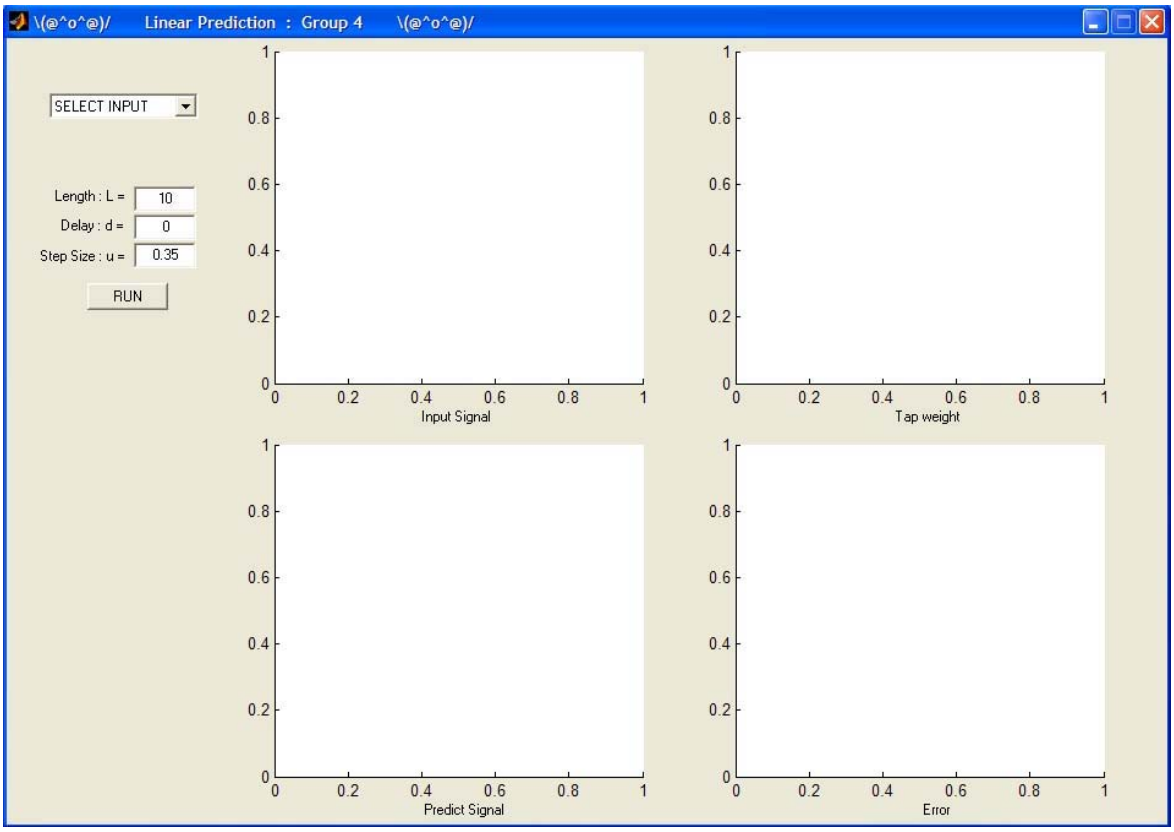
Summary

From experiment will show both random noise and speech have the same result. If delay is not zero, error will not converge. But if delay is zero, error will converge. if μ is changed, error will change. In the case of μ, if μ is a large number, the system will have a small error. But if μ is a small number, the system will have more error. In the case of L, if L is a small number, the system will have a small error. But if L is a large number, the system will have more error.

Linear prediction using NLMS algorithm can not use in the case have delay in the system.

GUI (Graphic User Interface)

In part of GUI can be access by type linear in command window then GUI show following



First choose input signal for simulation (random noise or speech) then set parameter follow this, length of tap weight (L), delay (d) and step size (u), which initial value is L= 10, d= 0 and u=0.35. And press RUN button this program will be simulate linear prediction using NLMS algorithm, wait few second its will show result in GUI consist Input signal, Prediction signal, value of Tap weight and Error .

Name		Code	Responsibility
1. Mr.Khaimtana	Suwapichpoom	4570233021	MATLAB GUI, Report, Web page
2. Mr.Chalotorn	Chanasong	4570275321	MATLAB programming, Present
3. Mr.Wattana	Keyoonratsami	4570528121	MATLAB programming, Report
4. Mr.Sunchai	Deelertpaibool	4570583621	Report, Present